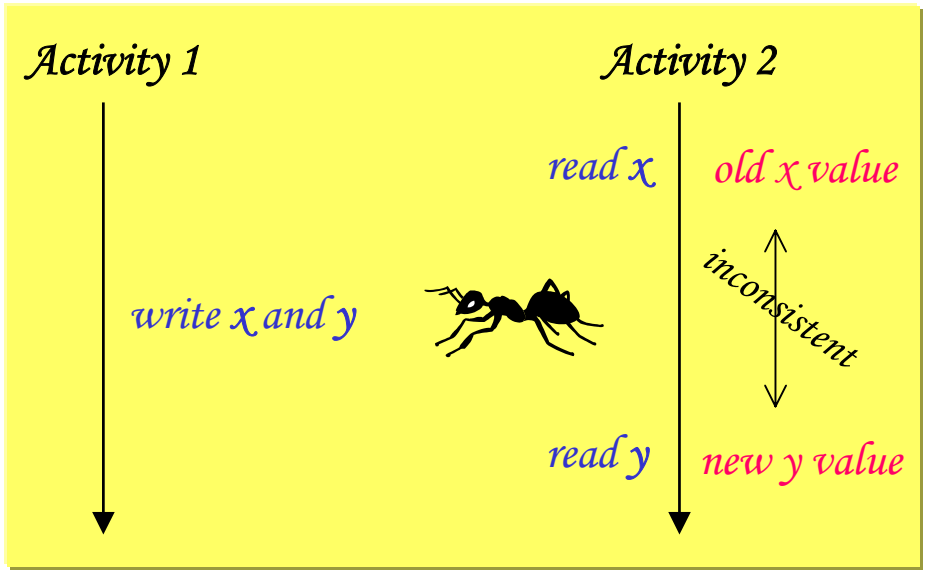
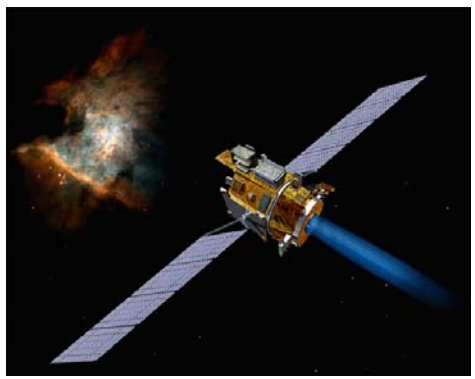
High-Level Data Race Detection



Problem: High-Level Data Races occur when different activities access shared resources with different atomicity views. As an example (left), activity 1 writes to x and y atomically in between activity 2 reading these values (non-atomically). Therefore activity 2 gets inconsistent values for x and y.

Solution: An algorithm has been designed and implemented that can examine an execution trace to find view inconsistencies. For example, activity 1 has the view {x,y} while activity 2 has the views {x} and {y}, which are regarded as inconsistent with {x,y}. The algorithm scales to large programs and is completely automatic.



Relevance: The Deep-Space 1 Remote Agent had a high-level data race that was very hard to find. This could have caused space craft state inconsistencies to be ignored by executing tasks during flight.

Explanation of Accomplishment

- **POC:** Klaus Havelund (ASE group, Code IC, havelund@email.arc.nasa.gov)
- **Collaborators:** Cyrille Artho (ETH Zurich, Switzerland)
- **Background:** Concurrency-related errors in multi-threaded mission software often manifest themselves only under rare circumstances, sometimes never during test, but in flight. Low-level data races is an example. Low-level data races occur when several activities access a shared resource simultaneously, causing the resource to have an inconsistent state. Algorithms exists for detecting such data races. We have gone beyond and studied High-level data races. High-level data races occur when different activities, that execute in parallel, access shared resources, but with different atomicity views. This may cause such activities to obtain inconsistent snapshots of the shared resources. A high-level data race was for example detected in the Deep-Space 1 Remote Agent before flight, using model checking, however requiring substantial manual effort.
- **Accomplishment:** We have developed a runtime analysis algorithm for high-level data race detection. The algorithm detects inconsistencies in the views that different activities have on shared resources. The algorithm works by analyzing a single randomly chosen execution trace for operations that take and release locks and for operations that access shared resources. From this information can be calculated whether all activities have consistent views. Inconsistent views typically arise if at least one activity "does it right". A paper with the title "High-Level Data Races" has been accepted for presentation at VVEIS'03 (The First International Workshop on Verification and Validation of Enterprise Information Systems). The paper appears to be the first on this subject in the scientific literature. The algorithm has been applied to the K9 Rover, developed at NASA Ames and to 2 other industrial applications. An inconsistency was detected in the K9 rover, although non-critical.
- **Future Plans:** We plan to improve the algorithm by studying its application to more case studies. The relationship to database concurrency and hardware concurrency theory will be investigated. It appears that perhaps techniques from these fields may apply to concurrency in programs with shared variables. For example, we will investigate how our algorithm relates to transaction serialization in database theory.